Symmetry-Aware Transformer Training for Automated Planning

Generalized policies, extrapolation, and planning symmetries

Markus Fritzsche Elliot Gestrin Jendrik Seipp
Linköping University

PRL Workshop @ ICAPS 2026

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM
LINKÖPING UNIVERSITY

Outline

Generalized planning task

Planning task

We write a task as $\Pi = \langle \mathcal{D}, \mathcal{I} \rangle$.

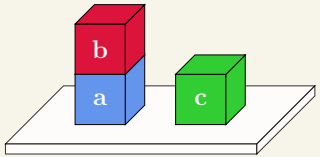
$$\mathcal{D} = \langle \mathcal{P}, \mathcal{A} \rangle, \qquad \mathcal{I} = \langle \mathcal{O}, s_0, G \rangle.$$

$$s \subseteq \{p(\bar{o}) \mid p \in \mathcal{P},\ \bar{o} \in \mathcal{O}^{\mathrm{arity}(p)}\}.$$

The domain is fixed; the instances vary in objects, initial states, and goals.

Example (Blocks instance)

$\mathcal{D}$: `on`, `clear`, `stack`, `unstack`, ...
$\mathcal{O} = \{a, b, c\}$
$s_0 = \{$`on(b,a)`, `clear(b)`, ...$\}$
$G = \{$`on(b,c)`, `on(a,b)`$\}$



Learning problem: generalized policies

Target

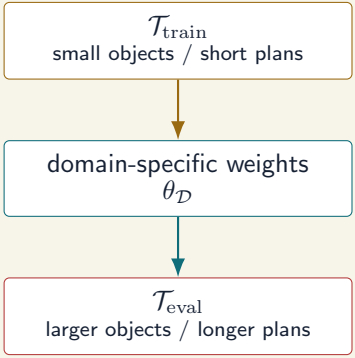
For each domain $\mathcal{D}$, learn weights $\theta_{\mathcal{D}}$ once and reuse them across instances:

$$\pi_{\theta_{\mathcal{D}}}(s, G) \in A(s)$$

or learn a value function

$$V_{\theta_{\mathcal{D}}}(s, G) \approx \text{cost-to-go}(s, G).$$

- Training: small instances from one domain.
- Primary question: **does the learned algorithm extrapolate?**



Interpolation stays near training sizes; extrapolation tests the generalized-policy claim.

Why Transformers are tempting

Plan generation

Planning is naturally sequence-valued:

$$(s_0, G) \mapsto \langle a_1, \ldots, a_k \rangle.$$

- Transformers are strong sequence models.
- Decoder models give direct plan generation.
- Encoder-decoder models separate input structure from action order.

But planning input is not text

The surface sequence is mostly an arbitrary encoding of:

- object identifiers,
- unordered sets of atoms,
- variable-size relational structure.

Self-attention: the relevant building block

For token embeddings $X \in \mathbb{R}^{N \times d}$:

$$Q = XW_Q, \qquad K = XW_K, \quad V = XW_V,$$

$$\alpha = \mathrm{softmax}(QK^\top / \sqrt{d_k}),$$

$$\mathrm{Att}(X) = \alpha V.$$

Key property

Without positional information, self-attention is permutation-equivariant: reordering tokens reorders hidden states in the same way.

$x_1 = \texttt{on(b,a)}$, $x_2 = \texttt{clear(b)}$, $x_3 = \texttt{goal-on}$

keys

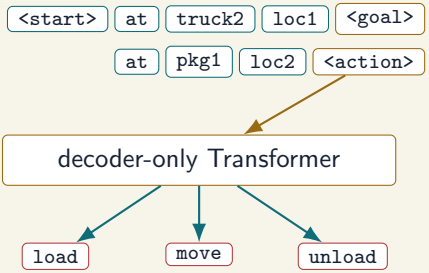
queries	x_1	x_2	x_3	
x_1	.15	.70	.15	◀ $h_1 = .15v_1 + .7$
x_2	.20	.65	.15	
x_3	.45	.20	.35	

Each row is a distribution over input atoms.

This property is useful for sets. Learned absolute positions deliberately break it.

PlanGPT: decoder-only plan generation

- GPT-2 style decoder-only Transformer.
- Input is a linearized state and goal.
- Output is generated autoregressively as action tokens.
- Objects are randomized into fixed type-specific vocabularies.



Training objective

Standard next-token cross entropy on complete plans.

The core mismatch: VisitAll

Semantic task

VisitAll has a grid, one robot position, and cells that must be visited.

same grid, many encodings

Two representation failures

1. **Renaming.** The cell name `pos-1-2` versus `v17` is arbitrary, but a policy must choose the corresponding move.
2. **Hard-coded positions.** If the model learns grid coordinates as vocabulary or absolute sequence positions, unseen cells in larger grids have no learned representation.

$\langle \mathcal{D}, \mathcal{I} \rangle$ **is not one serialized token string.**

Struggle 1: renaming

Same VisitAll state, two vocabulary assignments:

Mapping A

```
cell(1,1) -> p11,  cell(2,1) -> p21
at(p11), connected(p11,p21)
```

Mapping B

```
cell(1,1) -> v7,  cell(2,1) -> v3
at(v7), connected(v7,v3)
```

The correct action is the same semantic move, only renamed:

$$\mathrm{move}(p11, p21) \quad \leftrightarrow \quad \mathrm{move}(v7, v3).$$

Combinatorics

With $|\mathcal{O}|$ objects and $|\mathcal{V}|$ vocabulary names:

$$\frac{|\mathcal{V}|!}{(|\mathcal{V}| - |\mathcal{O}|)!}$$

equivalent assignments.

Example (Eight cells)

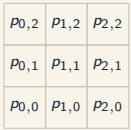
$$\frac{8!}{(8-8)!} = 40320$$

equivalent encodings.

Struggle 2: hard-coded grid positions

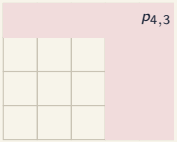
If grid structure is encoded as names or absolute positions, extrapolation needs tokens never trained.

Training grid



`p00, p01, ..., p22`

Extrapolation grid



`p30, ..., p43` are new.

Two related problems

new grid cells and longer serialized states require unseen absolute positions.

Design principle

Exploit what attention already gives us

Encoder self-attention without positional encodings is permutation-equivariant.

- Use an encoder for unordered state and goal atoms.
- Encode each atom as one compositional embedding.
- Use goal-predicates instead of a positional goal separator.

Example (Atom embedding)

For arity bound m:

$$T_{p(o_1,\ldots,o_n)} = W(E[p] \mid E[o_1] \mid \cdots \mid E[\mathrm{pad}]) + b.$$

Example (Goal atom)

`at(truck3, loc2)` becomes `goal_at(truck3, loc2)`.

Contrastive training for object names

For each training sample, create two renamed copies:

$$X,\ X' \quad \text{and} \quad Y,\ Y'.$$

They describe the same underlying planning structure.

$$L_{\text{att}} = \frac{1}{B} \sum_{\alpha \in \text{Att}} \|\alpha - \alpha'\|^2$$

$$L_{\text{hid}} = \frac{1}{B} \sum_{l} \|H^{(l)}_{[:d_k]} - H'^{(l)}_{[:d_k]}\|^2$$

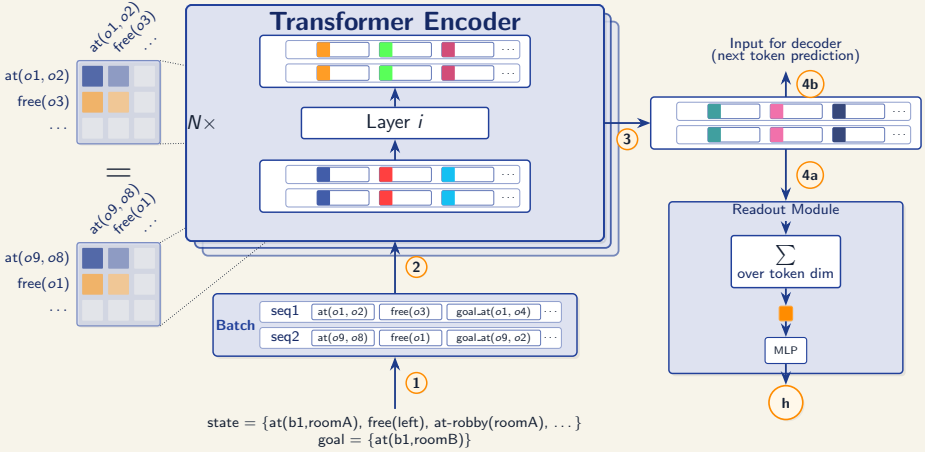
$$L = L_{\text{pred}} + L_{\text{att}} + L_{\text{hid}}.$$

Intuition

Same task, different names should induce:

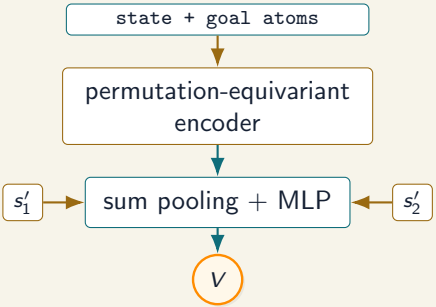
- corresponding attention patterns,
- corresponding name-independent hidden features,
- the same value or a renamed plan.

Encoder with contrastive objective



Two architectures

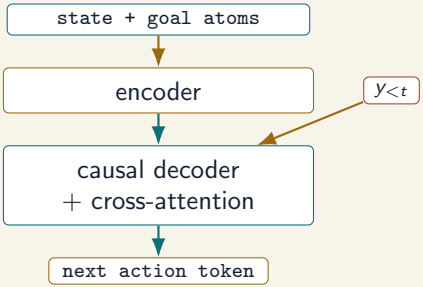
SymT$^{\mathrm{E}}$: value model



$$\pi_{\theta_{\mathcal{D}}}(s, G) = \arg\min_{a \in A(s)} V_{\theta_{\mathcal{D}}}(s_a, G)$$

Strength: compact value model; loss matters most here.

SymT$^{\mathrm{ED}}$: plan generator



$$y_t = \arg\max_{w \in \mathcal{V}} P_\psi(w \mid y_{<t}, E_\phi(s, G))$$

Strength: strongest coverage; explicit action-order model.

Applicability filtering: segmented action prediction

Current Blocks state

```
on(a,b)
on(c,d)
clear(a)   clear(c)
on-table(b)   on-table(d)
handempty
```

Raw next-segment scores

prefix: `[unstack,...]`

next token	p	legal?
unstack	.28	**no**
b	.24	**no**
a	.21	yes
c	.17	yes
d	.10	no

Greedy picks an invalid continuation.

After applicability filtering

legal arg_1: clear and stacked

next token	p_{filtered}
unstack	.00
b	.00
a	.21
c	.17
d	.00

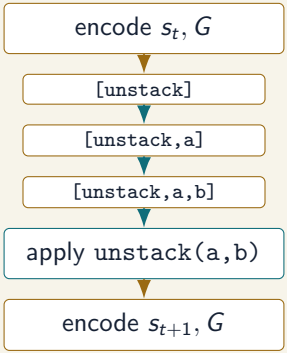
selected prefix: `[unstack, a, ...]`

Next segment after `[unstack,a,...]` filters to the unique support token `b`, yielding `unstack(a,b)`.

Regrounding: apply the completed action

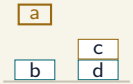
Before: s_t

```
on(a,b)
on(c,d)
clear(a)   clear(c)
on-table(b)   on-table(d)
handempty
```



After: s_{t+1}

```
holding(a)
clear(b)
on(c,d)
clear(c)
on-table(b)   on-table(d)
```



State update

delete: `on(a,b)`, `clear(a)`, `handempty`
add: `holding(a)`, `clear(b)`

Where the symmetry-aware loss matters

Encoder-only is the stress test

In VisitAll and Logistics, the encoder-only value model did not even converge in training loss without the symmetry-aware objective.

Empirical pattern

The loss helps both architectures, but it is most decisive for SymT^{E}.

Hypothesis

The difficulty is content-dependent: information must move between atoms through chain-like reasoning before the model can assign a useful value.

Curriculum

Training first on short distances and then larger ones helped, but was unstable across runs.

Coverage: extrapolation is the point

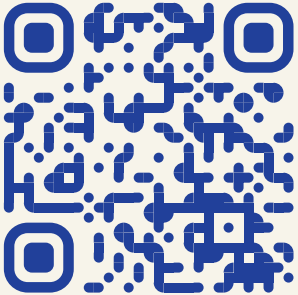
		PlanGPT - Decoder (baseline)			SymT^{E} (ours)	SymT^{ED} (ours)		
		greedy	applicable	regrounding	greedy	greedy	applicable	regrounding
Blocks	validation	.00±.00	.00±.00	.00±.00	**1.00±.00**	**1.00±.00**	**1.00±.00**	.00±.00
	interpolation	.56±.16	.56±.16	.00±.00	**1.00±.00**	**1.00±.00**	**1.00±.00**	**1.00±.00**
	extrapolation	.00±.00	.00±.00	.00±.00	.05±.07	.07±.02	**.13±.05**	.00±.00
Gripper	validation	.00±.00	.00±.00	.00±.00	**1.00±.00**	.17±.24	**1.00±.00**	**1.00±.00**
	interpolation	.00±.00	.44±.16	.00±.00	.89±.16	.67±.00	**1.00±.00**	**1.00±.00**
	extrapolation	.00±.00	.00±.00	.00±.00	.02±.03	.00±.00	.15±.06	**.79±.16**
Visitall	validation	.00±.00	.14±.12	.00±.00	**1.00±.00**	.33±.09	.93±.04	.99±.02
	interpolation	.05±.04	.67±.18	.41±.22	**1.00±.00**	.87±.01	.99±.01	**1.00±.00**
	extrapolation	.00±.00	.02±.02	.00±.00	.42±.11	.00±.00	.15±.05	**.64±.12**
Logistics	validation	.00±.00	**.08±.12**	.00±.00	.00±.00	.00±.00	.00±.00	.00±.00
	interpolation	.07±.05	**.44±.09**	.19±.14	.11±.00	.22±.31	.26±.29	.22±.31
	extrapolation	**.00±.00**	**.00±.00**	**.00±.00**	**.00±.00**	**.00±.00**	**.00±.00**	**.00±.00**

Takeaways from the results

- PlanGPT solves some interpolation cases but essentially fails on larger extrapolation instances.
- SymT$^{\mathrm{ED}}$ gives the strongest coverage overall.
- SymT$^{\mathrm{E}}$ shows that Transformer encoders can act as learned value functions.
- The symmetry-aware loss is most visible for SymT$^{\mathrm{E}}$: without it, VisitAll and Logistics can fail to converge.
- Logistics remains hard; chain-like information propagation is still a bottleneck.

Conclusion

- The central obstacle is not planning syntax; it is **symmetry**.
- Atom-order equivariance belongs in the architecture.
- Object-name equivariance can be encouraged directly in the loss.
- Transformers improve substantially, but they still do not fully match GNN-style generalization.



Thank you.

Questions & discussion

marfr65@liu.se
Linköping University

Backup slides

appendix

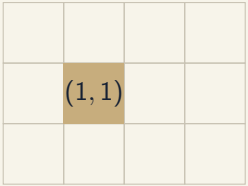
Appendix: names can leak structure

Visitall example

Location objects often carry coordinates:

$$\texttt{loc-x1-y2} \quad \rightsquigarrow \quad \langle 1, 2 \rangle.$$

- Keeping semantic names makes the model look better.
- It can memorize coordinate-like strings.
- Generalization to unseen grid sizes becomes ambiguous evidence.



loc-x1-y1

We randomize all object names.

Appendix: experimental setup

- Domains: Blocks, Gripper, Visitall, Logistics.
- Train per domain/architecture for 12 hours on A100.
- Three random seeds per configuration.
- Baseline: PlanGPT with all object names randomized.

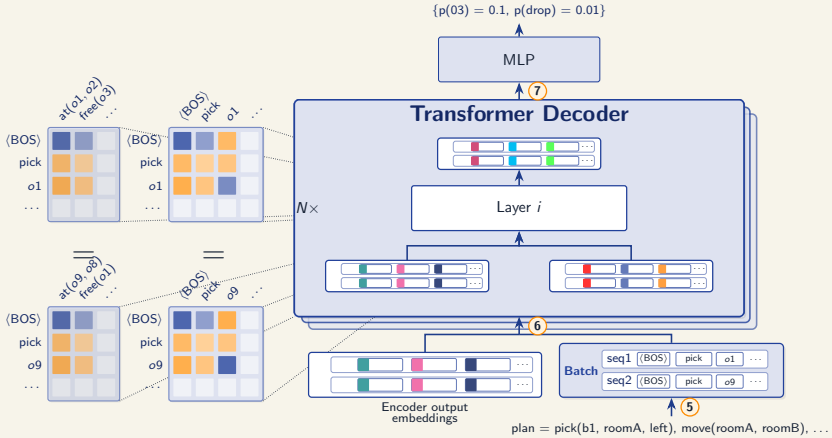
Evaluation splits

- validation: slightly larger than training;
- interpolation: comparable sizes, distinct tasks;
- extrapolation: larger than validation.

Metric

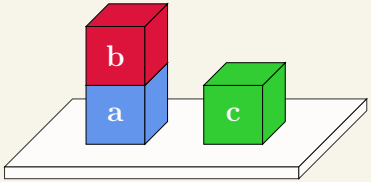
Coverage: fraction of tasks for which a valid plan is generated.

Appendix: decoder



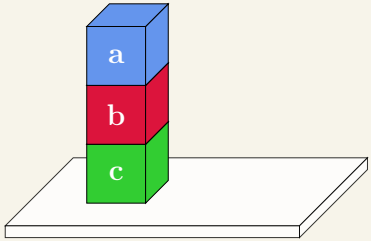
Appendix: original Blocks example

Initial state




```
on-table(a)
  on(b, a)
on-table(c)
  clear(b)
  clear(c)
```

Goal state




```
on(b, c)
on(a, b)
```

Appendix: PlanGPT limitations

1. **Object assignment equivariance:** object names are arbitrary.
2. **Information leakage:** semantic names can leak structure.
3. **Atom order equivariance:** $|s_0|! \cdot |G|!$ equivalent orderings per assignment.
4. **Learned positional encodings:** weak extrapolation to unseen positions.